Appendix A : Markov chain Monte Carlo (MCMC) algorithms.

This appendix discusses the MCMC algorithms used to sample from $P(\mathbf{\Theta}, \nu_1, .., \nu_N | \mathbf{Y}_1, ..., \mathbf{Y}_N)$ for the MCKL method, the kernel density estimator for the MCKL method, and calculation of final likelihood values after MCKL estimation of the MLE. First I describe an MCMC algorithm for $P(\nu|\mathbf{Y}, \mathbf{\Theta})$ for a single replicate, and then I describe the algorithm for $P(\mathbf{\Theta}, \nu_1, .., \nu_N | \mathbf{Y}_1, ..., \mathbf{Y}_N)$. For each I describe example Metropolis-Hastings sampling steps in detail and then list the full combination of sampling steps used.

To set the basic concepts (Gilks et al. 1996; Robert and Casella 1999), consider a Metropolis-Hastings MCMC algorithm to sample from the distribution $P(A)$. Let A_i and A_{i+1} be the current and next values of A. MCMC works by considering a random proposal value, A', and using an acceptance probability to decide whether $A_{i+1} = A'$ or $A_{i+1} = A_i$. Specifically, if the proposal density is $q(A'|A_i)$, the acceptance probability is:

$$P_{\text{accept}} = \min\left(1, \frac{P(A')q(A_i|A')}{P(A_i)q(A'|A_i)}\right) \tag{A.1}$$

Metropolis-Hastings algorithms for different dimensions or blocks of dimensions of A can be used iteratively or randomly to produce a chain with stationary distribution $P(A)$.

MCMC algorithm 1

For sampling from $P(\nu|\mathbf{Y}, \mathbf{\Theta})$ (omitting the j subscript), write the components of ν as $\nu = (\beta, \mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_{20})$ where $\mathbf{z}_t = (z_{t,E}, z_{t,J}, z_{t,A})$. To sample from $\mathbf{z}_t$, draw a proposal $\mathbf{z}'_t$ from $q(\mathbf{z}'_t|\mathbf{z})$, here a normal proposal distribution with mean $\mathbf{z}_t$ and covariance matrix $\sigma_q^2 I$, where I is the 3x3 identity matrix and $\sigma_q = 0.5$. Let ν' be the noise vector with this proposal value, $\nu' = (\beta, \mathbf{z}_1, .., \mathbf{z}'_t, .., \mathbf{z}_{20})$. Then the acceptance probability is:

$$P_{\text{accept}} = \min\left(1, \frac{P(\mathbf{Y}|\nu', \mathbf{\Theta})P(\mathbf{z}'_t|\mathbf{\Theta})}{P(\mathbf{Y}|\nu, \mathbf{\Theta})P(\mathbf{z}_t|\mathbf{\Theta})}\right) \tag{A.2}$$

In this case, the ratio of "backward" to "forward" proposals, $q(\mathbf{z}_t|\mathbf{z}'_t)/q(\mathbf{z}'_t|\mathbf{z}_t)$, from (1) (with $A = \mathbf{z}_t$) is always 1 and has been omitted. The probabilities of β and the other $\mathbf{z}$ values are the same for ν and ν', so their probabilities also cancel. To complete this sampling step, a uniform random variable is drawn, and $\mathbf{z}'_t$ is accepted if the uniform draw is less that P_{accept}. Otherwise $\mathbf{z}'_t$ is rejected and the next value in the Markov chain is the same as the previous value, $\mathbf{z}_t$.

To obtain sampling steps for β, I used a lognormal sampling distribution with the log of the proposal centered on the log of the current value:

$$q(\beta'|\beta) \propto \frac{1}{\beta}\exp[-\frac{(\log(\beta') - log(\beta))^2}{2\sigma_q^2}] \tag{A.3}$$

where for this step $\sigma_q = 0.15$. The acceptance probability for this step is calculated similarly to (2), with β and β' in place of $\mathbf{z}$ and $\mathbf{z}'$, but in this case the proposal distribution is not symmetric, so the ratio of "backward" to "forward" proposals must be included:

$$P_{\text{accept}} = \min\left(1, \frac{P(\mathbf{Y}|\nu', \mathbf{\Theta})P(\beta'|\mathbf{\Theta})q(\beta|\beta')}{P(\mathbf{Y}|\nu, \mathbf{\Theta})P(\beta|\mathbf{\Theta})q(\beta'|\beta)}\right) \tag{A.4}$$

A more general way to think about this type of sampling step is that we can transform coordinates to work with $\log(\beta)$ instead of β, use a proposal distribution in those coordinates, and then account for the transformation back to our original coordinates when we calculate $P(\mathbf{z}'|\mathbf{\Theta})$, which yields the same acceptance probability. This approach will be used for the next MCMC algorithm.

Sampling steps for different components of ν can be combined in many ways. One useful approach is to sample from adjacent $\mathbf{z}_t$ values together. For example, to sample from five adjacent $\mathbf{z}_t$ values with $t = 1..5$, we could propose $\mathbf{z}'_t = \mathbf{z}_t + \eta$, where η is normally distributed with mean 0 and covariance matrix $\sigma_q^2 I$, i.e. use the same normal shift for several $\mathbf{z}_t$'s. We then calculate P_{accept} in the same way as (2).

One iteration of the full sampler for $(\nu|\mathbf{Y}, \mathbf{\Theta})$ consisted of samples from blocks of 10 $\mathbf{z}_t$ values ($t = 1..10$ and $11..20$), blocks of 4 $\mathbf{z}_t$ values ($t = 1..5, 6..10, 11..15, 16..20$), individual samples from each of the 20 $\mathbf{z}_t$ values, and 5 samples from β. In the MCMC literature, a sampling algorithm that efficiently moves around the target distribution is termed "well-mixed", and in the model here mixing β turns out to be especially important, which is why 5 sampling steps for β were used for each iteration.

MCMC algorithm 2

Metropolis-Hastings sampling in the MCKL algorithm for $(\mathbf{\Theta}, \nu_1, .., \nu_N|\mathbf{Y}_1, .., \mathbf{Y}_N)$ was considerably more complicated because there were strong correlations among the parameter dimensions and between the parameter dimensions and the process noise dimensions. MCMC algorithms that sample in one direction at a time can be inefficient if the sampling directions do not match the directions of correlation in the target distribution. An example of a correlation among parameter dimensions is that μ_β was negatively correlated with a_E because higher fecundity and lower survival can together produce similar population trajectories, and thus similar likelihoods, as lower fecundity and higher survival. An example of a correlation between parameter dimensions and process noise dimensions is that different process noises were more likely (given the data) with higher fecundity and lower survival than with lower fecundity and higher survival.

To obtain good mixing, I used a more complex set of Metropolis-Hastings steps than in MCMC Algorithm 1. A first change was that for $\mathbf{\Theta}$, I used a parameterization different than that of the text. Instead of a_E, a_J, and a_A, I used $S_E = S_E(0)$, $S_J = S_J(0)$, and $S_A = S_A(0)$. Second, working from this parameterization, I combined a number of sampling steps that move in directions of parameters and/or states that are motivated by biological interpretations of the model. An example to be explained in detail is of sampling S_E and $(\beta_1, \beta_2, .., \beta_N)$ together. This makes sense because a shift in S_E changes the survival values for each replicate, so a matching shift in fecundity values can produce more likely state trajectories. A biologically-motivated way to do this is to draw a random proposal value of S_E and then choose β_i values to hold constant the number of eggs that would mature to juveniles in the mean environment: $k_i = \beta_i S_E^{L_E}$. Given a proposal value S'_E, k_i can be kept constant by choosing the proposal β value $\beta'_i = \beta_i(S_E/S'_E)^{L_E}$, for each i. However, this adjustment affects the Metropolis-Hastings algorithm and must be accounted for as follows.

View $(S_E, k_1, ..., k_N)$ as a coordinate transformation, $(S_E, k_1, .., k_N) = g(S_E, \beta_1, .., \beta_N)$. The proposal probability in the transformed coordinates involves only the S_E dimension. The proposal probability in the original coordinates is given by standard probability theory as:

$$P(S_E, k_1, .., k_N) = \frac{P(S_E, \beta_1, .., \beta_N)}{|g'(S_E, \beta_1, ..., \beta_N)|} \tag{A.5}$$

where $g'(S_E, \beta_1, ..., \beta_N)$ is the Jacobian matrix of derivatives of g with respect to each of its arguments, and $|g'|$ is the determinant of the Jacobian. In this example, the Jacobian has non-zero values only on the diagonal and on the left-most column. The diagonal values are $(\partial S_E/\partial S_E = 1, \partial k_1/\partial \beta_1 = S_E^{L_E}, .., \partial k_N/\partial \beta_N = S_E^{L_E})$. The left-most column elements drop out because the determinant of this matrix structure reduces to the product of the diagonal elements.

Now the acceptance probability is:

$$P_{\text{accept}} = \min(1, \frac{[\prod_{i=1}^N P(\mathbf{Y}_i|\nu'_i, \mathbf{\Theta}')]P(S'_E)[\prod_{i=1}^N P(\beta'_i|\mathbf{\Theta}')]S_E^{NL_E} q(S_E|S'_E)}{[\prod_{i=1}^N P(\mathbf{Y}_i|\nu_i, \mathbf{\Theta}')]P(S_E)[\prod_{i=1}^N P(\beta_i|\mathbf{\Theta})]S_E'^{NL_E} q(S'_E|S_E)}) \tag{A.6}$$

where as before prime indicates proposal values and no prime indicates current values. For a proposal distribution $q(S'_E|S_E)$, I used a "reflected normal" proposal, which is obtained by adding a normal random variable to S_E and "reflecting" about 0 or 1, since we must have $0 < S_E < 1$. "Reflecting" means that, for example, if $S_E = 0.95$ and the normal random variable is 0.07, instead of $S'_E = 1.02$ we would use $S'_E = 0.98$. (In theory one might need multiple reflections, but in practice that didn't arise for the proposals used here.) With reflections it still turns out that $q(S_E|S'_E)/q(S'_E|S_E) = 1$.

With this general approach to constructing Metropolis-Hastings steps in temporarily transformed coordinate spaces, with the transformation accounted for by a Jacobian rule like (5), one can easily use steps that take advantage of biological interpretations of the model. Of course there are simpler approaches to obtain valid MCMC samplers, but the steps used here greatly improve efficiency over sampling one dimension at a time. The Metropolis-Hastings steps I used for a full sampling iteration were: each of b_E, b_J, and b_A in log coordinates; S_E with each $\beta_j S_E^{L_E}$ held constant by changing β_j; S_J with each $\beta_j S_J^{L_J}$ held constant; S_E and S_J (both randomly sampled) with each $\beta_j S_E^{L_E} S_J^{L_J}$ held constant; S_E and b_E (both randomly sampled with b_E in log coordinates) with each $\beta_j S_E^{L_E}$ held constant; S_A with each $\beta_j/(1 - S_A)$ (lifetime reproductive success) held constant; L_E in $\log(L_E - 2)$ coordinates; L_J in $\log(L_J - 2)$ coordinates; $L_E + L_J$ holding $L_E - L_J$ and each $\beta_j S_E^{L_E} S_J^{L_J}$ constant; μ_β in log coordinates; σ_β in log coordinates; and all process noises sampled with the same combination of steps described above for MCMC Algorithm 1.

The state of $(\mathbf{\Theta}, \nu_1, .., \nu_N)$ was saved after every 15 iterations, until a sample of $M = 10000$ states had been obtained. (Implemented in GNU C++ with a 2.0 GHz Pentium processor, each run takes approximately 1 hour.) Finally, the "priors" for the parameters were: $N(\mu = 4, \sigma = 100)$ for μ_B (although this is centered on 4, it is essentially flat and provides no prior information); exponential distribution with mean 100 for σ_β, L_E, L_J, b_E, B_J, and b_A; and uniform distribution between 0 and 1 for S_E, S_J, S_A.

Kernel Density Estimation

For MCKL, samples $\{\mathbf{\Theta}_j\}$ were reparameterized into standardized principal components before maximization. This linear transformation gives approximately independent, similarly scaled coordinates. In general, reparameterization of $\mathbf{\Theta}$ after sampling but before kernel density maximization requires consideration of Jacobians of the parameter transformation (as in equation 5). However, the transformation to principal coordinates is linear, so the Jacobian is constant and MCKL maximization is unaffected.

The kernel K_h was multivariate Gaussian and independent along the principal compenent axes. The smoothing bandwidth h in each direction was 0.66 and 0.79 for null and alternative hypotheses, respectively. These values were chosen so that for a unit normal Gaussian likelihood – approximately the shape of the principal components sample – and MC sample size $M = 10000$, there would be approximately a 95% chance of obtaining an estimate with likelihood ≥ 0.95 of the true maximum, which for log-likelihood-based hypothesis testing is quite accurate.

Final Likelihood values

The MCKL method gives a maximum likelihood known only up to the unknown constant C_P. To estimate the actual likelihood at the MLE, I used MCMC Algorithm 1 to obtain a sample from $(\nu_i | \mathbf{Y}_i, \mathbf{\Theta})$ for each i, estimated the mean and covariance matrix of this sample, and used a multivariate normal distribution with that mean and covariance as an importance sampling distribution to calculate the likelihood. This importance sampling calculation is similar to equation (10) of the main text, with a different sampling distribution, $P_S(\nu_i)$, for each i. For MCMC Algorithm 1, process noises were saved after every 5 sampling iterations until a sample of 2000 was obtained to estimate the mean and covariance for the sampling distribution, and then a sample of 10000 points from that distribution was used for the likelihood estimate.

Literature cited

W. R. Gilks, S. Richardson, and D. J. Spiegelhalter (eds.) 1996. Markov Chain Monte Carlo in Practice. Chapman & Hall, New York, USA.

Robert, C. P. and Casella, G. 1999. Monte Carlo Statistical Methods. Springer, New York, New York, USA.